Siege前后端接口

传输方式：POST； 数据格式：JSON；

1. 注册

* 功能：玩家注册
* URL：<http://localhost:8088/register>
* 输入：（key: value，key统一为String类型）
* **“register”: bool** true时进行注册，false不进行注册，默认true
* 返回：（返回值类型均为String）
* **"register failed"** 注册失败
* **“{“address”: “xxx”, “publicKey”: “xxx”, “privateKey”: “xxx”, “privateKeyEncrypted”: false}”** 注册成功

1. 登陆

* 功能：玩家登陆
* URL：<http://localhost:8088/login>
* 输入：
* **“address”: String** 玩家地址
* **“publicKey”: String** 玩家公钥
* **“privateKey”: String** 玩家私钥
* **“privateKeyEncrypted”: bool** 私钥是否加密，默认为false
* 返回：
* **"login success"** 登陆成功
* **"login failed"** 登陆失败

1. 进入匹配，开始游戏

* 功能：将玩家加入匹配池，达到指定人数后开始游戏，初始化数据表
* URL：<http://localhost:8088/startGame>
* 输入：
* **“playerAddress”: String** 玩家地址
* **“signature”: String** 玩家签名 数据结构如下：

“{“address”: “xxx”, “publicKey”: “xxx”, “privateKey”: “xxx”, “privateKeyEncrypted”: false}”

* 返回：
* **"match success"** 匹配成功，所有玩家进入下一阶段
* **"already in match queue"** 该玩家已经在匹配队列中（为了防止用户掉线后重新匹配）
* **"match waiting"** 玩家匹配中
* **"match error"** 玩家匹配出错
* **"unknown error"** 未知错误
* **"transfer failed"** 缴纳入场费失败
* **"startGame failed"** 开始游戏失败

（后续逻辑部分还没定）

=========下面是关于区块链上数据表查询的前后端接口=========

1. 查询玩家信息表

* 功能：返回玩家游戏数据
* URL：<http://localhost:8088/getPlayerTable>
* 输入：（key: value，key统一为String类型）
* **“playerAddress”: String** 玩家地址
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名，数据结构同上
* 返回：（key: value，key统一为String类型）返回数据结构可能是JSONObject或者是JSONArray
* 暂时保存在我电脑上，等电脑修好①

1. 查询游戏中城池信息表

* 功能：返回每一场匹配中城池数据
* URL：<http://localhost:8088/getCitiesTable>
* 输入：
* **“gameId”: long** 游戏匹配的Id
* **“cityId”: long** 城池Id
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

（后续部分还没实现）

=========下面是关于区块链上用户资产查询、操作的前后端接口=========

1. 创建游戏道具

* 功能：项目方创建游戏资产，当前版本权利只交给项目方，玩家不可以创建资产，但是可以获取游戏资产
* URL：<http://localhost:8088/createGameItem>
* 输入：
* **“initialSupply”: long** 资产发行量
* **“uri”: String** 统一资源标志符，用于解析链上资产，同类资产仅有一个id和uri（后续资产和uri的对应表我会定下来，参考模板为“uri://siege-class-name-id”）
* **“signature”: String** 项目方签名
* 返回
* 待定

1. 分发游戏资产

* 功能：玩家无法创建游戏资产，通过游戏可以获取游戏资产，由项目方进行发放。
* URL：<http://localhost:8088/issueGameItem>
* 输入：
* **“id”: long** 资产在链上的Id
* **“to”: String** 接收方地址
* **“value”: long** 数量
* **“signature”: String** 项目方签名，资产直接从项目方地址转至to地址
* 返回：
* 待定

1. 游戏资产游戏内转移

* 功能：将玩家A中的资产1转至玩家B地址中。
* URL：<http://localhost:8088/transferGameItem>
* 输入：
* **“from”: String** 玩家A地址
* **“to”: String** 玩家B地址
* **“id”: long** 资产在链上的Id
* **“value”: long** 数量
* **“data”: String** 交易附言，可为空
* **“signature”: String** 地址from的签名
* 返回
* 待定

1. 查询某一资产数量

* 功能：查询指定id资产的余额
* URL：<http://localhost:8088/balanceOfGameItem>
* 输入：
* **“owner”: String** 拥有者地址
* **“id”: long** 资产在链上的Id
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

1. 查询用户所有资产

* 功能：查询某一用户所拥有的所有资产
* URL：<http://localhost:8088/propertyOfGameItem>
* 输入：
* **“owner”: String** 拥有者地址
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

1. 获取资产uri

* 功能：通过资产id获取对应的uri
* URL：<http://localhost:8088/getUriGameItem>
* 输入：
* **“id”: long** 资产在链上的id
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

1. 获取资产id

* 功能：通过资产uri获取对应的id
* URL：<http://localhost:8088/getIdGameItem>
* 输入：
* **“uri”: String** 资产uri
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定